Privacy in the semantic web– Social Networks based on XMPP

Jan Torben Heuer <jan.heuer@uni-muenster.de>

December 13, 2008

1 Introduction

In the last years the static web has moved towards a socially interactive web. Since 2005 we often refer to this as the *web2.0* [1]. People collaboratively write articles in online encyclopedias like Wikipedia[1] or self-portray themselves with profiles in social networks like Myspace[2]. Within a social network, members can link with each other in order to create a personal network of friends. Often, the number of friends is a kind of "social status" and displayed on a person's profile page. According to [2] this community aspect is one of the reasons why social networks attract so many users.

In current social network applications we see a **lack of privacy**. If private data is shared with others across a web based social network then our privacy can be affected on different levels. The minimum privacy requirement is that private data must not be visible to everyone. Therefore most services provide a simple access control for content. Flickr for instance supports private or family photo albums. Recently some security issues arose in social network which allowed users to gain private data of other users. In [3] has been shown how to get limited control over the computer of Facebook users in order to perform a denial-of-service attack against another host. The authors also explained how malicious applications can access a private user-profile in order to copy the private data to a remote server. So another privacy requirement is that data is kept secure – a trust that our application must also gain, of course. Most important is the trust in the service provider regarding what he actually *does* with private information. Besides explicitly provided information like preferences there are also implicit information from user tracking – visited profiles, groups or advertisements. Mostly such data are used for market analysis and user specific advertising. Our privacy is the price we have to pay in order to use such free services. But the price may be too high. The issue is not the advertising but the continuous collecting of data. Information that have been gathered once cannot be reverted. The membership in a discussion group about certain diseases for example can prevent someone from getting hired if the employer gets this knowledge.

[1]`http://www.wikipedia.org`
[2]`http://www.myspace.com`

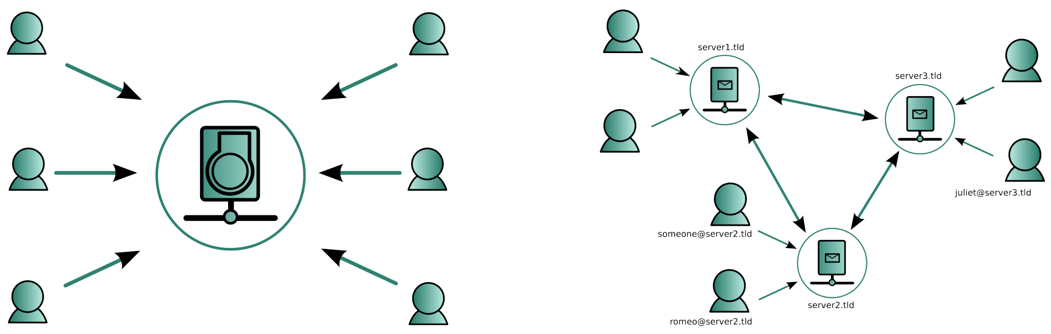
Moreover, such information can also cause disadvantages for later generations in case of genetic diseases. We are the first generation in the “information century” and long term impacts of the ongoing user profiling in social networks are not foreseen, yet. Social networks are a phenomenon which attracts a lot of users, especially users who do not have the technical knowledge to understand what happens in the background. Or what is possible with current user tracking software.

Additionally most content in social networks is kept in proprietary formats. There is no content integration between the existing independent social networks. All have their own community and neither user-accounts nor friendships can be shared between the social networks. Ideally data should be exchanged in an open and extensible format such as the Web Ontology Language. And here we we see the **conflict**. We cannot ensure privacy and ask for interoperability at the same time. On the one hand, an open exchange of content between different sites makes it easier to collect information about a user. An exchange of user profiles between social networks simplifies the aggregation and reasoning of hidden information. On the other hand, to ensure privacy protection the different social networks would have to restrict public access and use proprietary instead of open formats. This way, the vision of an open, interconnected and interoperable semantic web[4] won’t come true.

Regarding this conflict between interoperability and privacy we propose an alternative to traditional server-based social networks. In this paper we describe a distributed social network architecture and compare it to currently used server based applications. Like in traditional social networks users create an account which identifies them in the network. The users can create personal profiles and link to others in the network. Our architecture does not need a central server which stores private data of the users because personal data are shared directly with linked friends. Our application therefore does not have the privacy issues of social networks that we identified above.

2 A network architecture based on XMPP

In current web-based applications users connect to and store their (private) information on central servers. Figure 1(a) illustrates the principle design. The opposite of a client-to-server network is a *peer-to-peer* (P2P) network. In P2P networks, each peers are similar, which means that they can act as a server (data provider) and client (data consumer) at the same time. There is no central instance that routes the traffic or knows about all connected peers. Peer-to-peer technologies [5] are mainly used for anonymous filesharing where one peer tries to connect to as many other peers as possible. In contrast, the authors define a network where peers only communicate with manually selected peers as a *friend-to-friend* (*F2F*) network. We use a F2F network for our application because we only want communication between clients that *trust* each other. The authors additionally point out that a serverless network is not able to guarantee connectivity because clients are often located in a local network that is connected to the Internet through a router. In such configurations the router acts as a packet filter (often called *firewall*) with NAT (Network Address Translation). Two peers that are both located behind a NAT router cannot directly contact each other because connections must always be initiated from inside the local network to the outside (i.e. the Internet). For that reason a public available message relay is necessary where clients can initially connect to. The *extensible messaging and presence protocol XMPP* ([6]) is such a messaging network architecture for client-to-client communications (see figure 1(b)).



(a) Client-server architecture: a server stores the client's data in its database. Incomming queries are evaluated by the server.

(b) Messaging network: clients store their data, the servers are only responsible for communication. Clients directly query each other for information.

Figure 1: Differences in communication and data location

A XMPP network consists in a set of independent servers. The servers needn't know each other initially and anyone can set up his own XMPP server. Users register at their preferred server and get an user–id which is similar to an e–mail address: `user@server.tld`. Two clients which exchanged their addresses (and authorized each other for communication) can then send messages to each other. The servers only handle the message routing and presence information (if a user is currently online) but do not need to know about the concrete information that are exchanged. At the beginning we argued that server based social networks in the web cannot ensure privacy. Although we also rely on servers in our architecture their role is different. Servers are only responsible for transferring messages but needn't know about the message contents.

In order to provide a private message exchange for clients we use the public-private-key encryption PGP. Messages can be encrypted to protect them against eavesdroppers and messages can be signed to prevent forgery. This technology is already used in the XMPP context for encrypting instant messaging but it isn't widely used, yet. There are two steps which are important in order to establish a really secure connection: At first, a pair of a public and private key has to be created. This can be done automatically by our application when a new XMPP account is created. Second, public keys have to be send to friends and their public keys must be verified. If someone pretends to be a certain person and introduces himself with fake name there won't be a possibility to automatically verify the identity. Like in other social networks anyone can sign up with any name. Unless we have a central authority which maintains identities - and we clearly don't want to have such an institution - people have to manually identify each other. That means, they have to verify that a public key is really owned by the the person it claims to. If this check isn't done carefully the future communication cannot be assumed to be secure. Typically, the verification is perfomed by compareing the fingerprint of the public key either by telephone or face-to-face. Our application can only support the users by i.e. asking for a friends fingerprint but we do not want to include any automation.

3 Data storage and exchange

In the article "The Semantic Web" [4] the authors described their vision about the future of the web. While the current web is for the people, the semantic web shall allow agents to read, understand and process the available information in order to support people in information retrieval and decision making. Although we're still far away from this vision becoming true, a lot of work has been done in recent years to archive the goal. The World-wide-web Consortium W3C developed the resource description framework RDF which is a general tool to model information in a structured way. Information are written as statements or triples of `{Subject, Predicate, Object}`, for example "user1 hasName Bob". The triples can be read by agents and processed by logic-interpreters (reasoners). Other popular data models are relational databases or XML documents. While relational databases store data in a tabular structures or XML documents have a tree structure, triples are graphs. RDF graphs for example do not have a root and can contain circular referenes. Each statement's subject can also be an object in another statement, for example "user1 knows user2".

Ideally, information are stored in a semantic format, distributed over the web and linked to each other. Today, the web concentrates at some websites. Most web based social networks do not provide an export of user data or have their own proprietary format or API. Although this formats are often structured with open standards like XML their content isn't interoperable. Each social networks defines its own structure. Our goal however is to directly share information between clients without the need to convert them manually. Also we don't want to bound the user to our application just because he doesn't get his data out of it. Therefore we use existing RDF schemas to model the user's information. The main schema is the FOAF vocabulary [7] which is used to share personal data like name, e-mail address and links to other friends. A graph of the vocabulary is shown in figure 2.

As an initial example we implemented a social semantic tagging application. Users can create their own personal profile and bookmark websites. The bookmarks can be annotated with keywords, so called *tags*. For this *tagging* already exists a RDF schema, the TagOntology [8]. Further extension can define their own schemas and connect it to the existing ones. If one wants to query information from his friends, i.e. their names and e-mail addresses or bookmarks for a certain tag he can use SPARQL [9] queries. The syntax is similar to SQL but for RDF statements instead of relational databases. This generic query approach allows for an easy retrieval of any resource. Of course, each user can define which personal data in his database are available to whom.

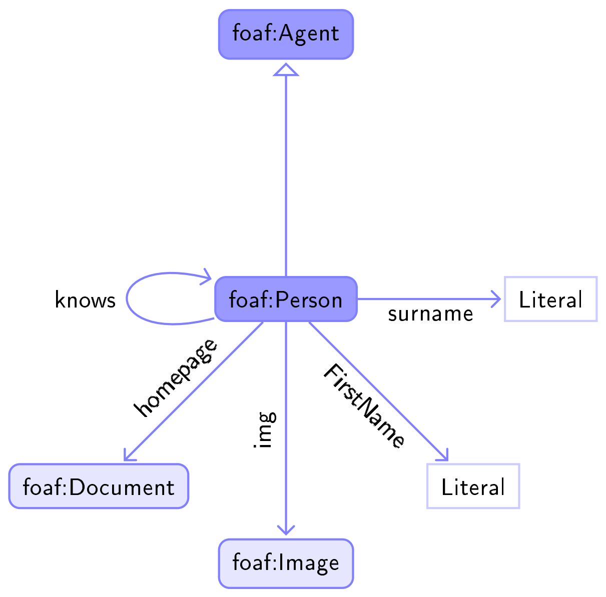
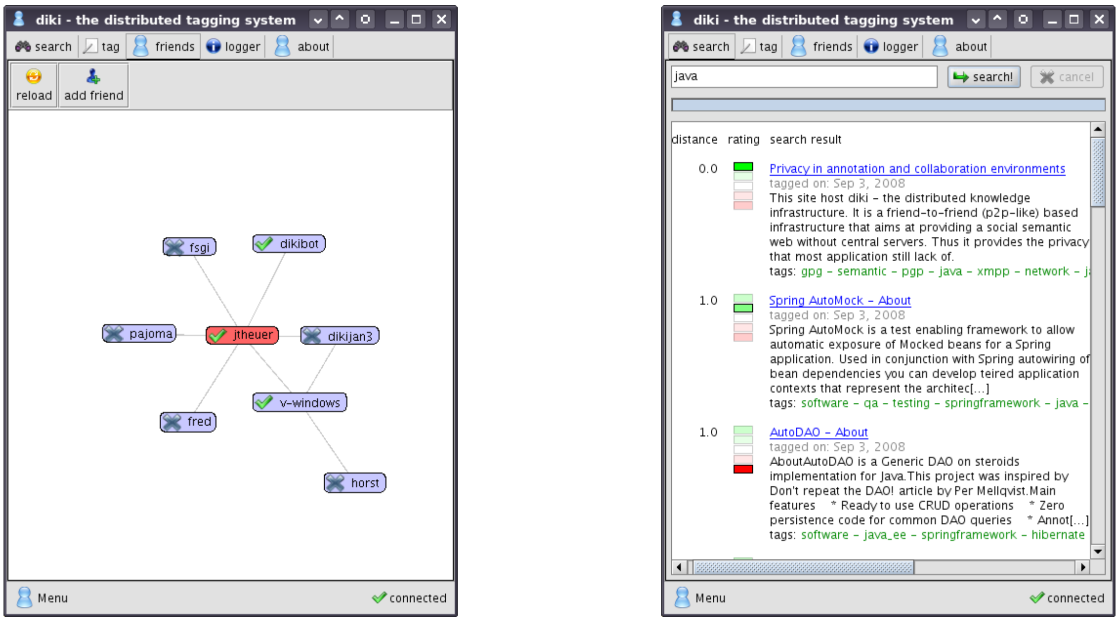


Figure 2: The classes Agent and Person from the FOAF vocabulary and some properties as graph. Person instances can reference to other Persons with the `knows` property. There are much more properties defined, i.e. to describe Jabber or OpenID accounts.

4 Application

We developed a first desktop application in Java for sharing bookmarks with friends. The latest Java release (Java 6) has a much better cross-platform desktop support than the releases before, for example “System Tray Icons” and access to the system–default web browser. Another reason for Java 6 was the Java Webstart technology which allows for automatic update detection on startup. The application can directly be started from the website `http://www.pace-project.org` which has been created for this project. However we want to stress out that our application is only one possible implementation of the concept. It might also be interesting to include the functionality in exiting instant messaging clients. The application can also run as a small server with a web frontend for the local browser so that it looks familiar to other web-based social networks. The main reason for an implementation in Java was that most functionalities are already provided by other projects. The XMPP protocol was implemented by the “Smack API” from igniterealtime[3]. PGP security functionalities are provided by the “Bouncy Castle Crypto APIs”[4]. For the database engine we use “Sesame” from Aduana[5]. It is a full-featured RDF store that directly supports SPARQL queries. In order to let people use their existing bookmarks we wrote two import plug-ins for Delicious and Bibsonomy. If a

[3]`http://www.igniterealtime.org/projects/smack/index.jsp`
[4]`http://www.bouncycastle.org/java.html`
[5]`http://www.openrdf.org/`



java

distance rating search result

0.0 Privacy in annotation and collaboration environments
tagged on: Sep 3, 2008
This site host diki – the distributed knowledge infrastructure. It is a friend-to-friend (p2p-like) based infrastructure that aims at providing a social semantic web without central servers. Thus it provides the privacy that most application still lack of.
tags: gpg – semantic – pgp – java – xmpp – network – j

1.0 Spring AutoMock – About
tagged on: Sep 3, 2008
Spring AutoMock is a test enabling framework to allow automatic exposure of Mocked beans for a Spring application. Used in conjunction with Spring autowiring of bean dependencies you can develop teired application contexts that represent the architec[...]
tags: software – qa – testing – springframework – java –

1.0 AutoDAO – About
tagged on: Sep 3, 2008
AboutAutoDAO is a Generic DAO on steroids implementation for Java.This project was inspired by Don't repeat the DAO! article by Per Mellqvist.Main features * Ready to use CRUD operations * Zero persistence code for common DAO queries * Annot[...]
tags: software – java_ee – springframework – hibernate

Figure 3: Two screenshots of the application. The social friends-graph (left) and a search result for the tag *java* (right).

user already has an account for the services he can link his bookmarks in to application.

Other locally running programs can interact with our application in order to trigger certain actions. The GUI provides access to its API with the integrated HTTP server jetty[6]. By clicking the “new Bookmark” button in Firefox for example, a HTTP request to our application is triggered: `http://localhost:30013/api/newBookmark?url=...`. The application opens the “new bookmark” window of our application and the currently displayed URL is filled in and can be annotated with tags. A similar plug-in also exists for the KDE web- and file browser “Konqueror”. The HTTP API can also provide additional services. Many people for example like to aggregate news in an RSS reader (RSS is a good example for a widely used RDF-based format). Sites like Delicious provide a “hotlist” which are the currently most often bookmarked sites. Our application can generate RSS feeds of the pages that have recently been bookmarked by friends.

5 Challenges and future work

Our application will only work if sufficient clients are online, like other client-based applications (i.e. for filesharing or instant messaging). Social networks however will only be attractive if a certain amount of interesting information are available (profiles of friends or bookmarks). Therefore we need a mechanism to cache data in the network. One possibility is that users can cache profiles of friends and distribute them while they

[6]`http://www.mortbay.org/jetty/`

are offline. But we have to make sure that neither the privacy of friends is affected (by giving information to the wrong people) nor that conflicts because of different versions occur. Another future goal of our application came up when we integrated PGP security. Currently, PGP public-keys are shared over keyservers. All data from this servers can be downloaded by everyone. There are for example keys, e-mail addresses and a set of users who signed the key. The singers are possible friends of the keyowner and therefore we can say that the keyserver's content is one of the first social networks[10]. But some people don't want to expose their information and therefore avoid the keyservers. Nevertheless keyservers are a convenient way to find and exchange public keys. We propose to use our architecture as a keyserver replacement. PGP keys for E-Mail communication are added to a user's FOAF profile and can be queried by friends only. The application provides a standard PGP keyserver interface on its default port for the localhost. Other applications like e-mail programs can then access keys of friends like before because internally the key search is translated to a SPARQL query in our application.

References

[1] Tim O'Reilly. What is web 2.0: Design patterns and business models for the next generation of software. *Social Science Research Network Working Paper Series*, September 2005.

[2] Adam Mathes. Folksonomies – Cooperative Classification and Communication Through Shared Metadata. `http://www.adammathes.com/academic/computer-mediated-communication/folksonomies.html`, December 2004.

[3] Elias Athanasopoulos, A. Makridakis, Spyros Antonatos, D. Antoniades, Sotiris Ioannidis, Kostas G. Anagnostakis, and Evangelos P. Markatos. Antisocial networks: Turning a social network into a botnet. In Tzong-Chen Wu, Chin-Laung Lei, Vincent Rijmen, and Der-Tsai Lee, editors, *ISC*, volume 5222 of *Lecture Notes in Computer Science*, pages 146–160. Springer, 2008.

[4] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific Amercian*, May 2001.

[5] Michael Rogers and Saleem Bhatti. How to disappear completely: A survey of private peer-to-peer networks. In *1st International Workshop on Sustaining Privacy in Autonomous Collaborative Environments (SPACE 2007), Moncton, Canada*, July 2007.

[6] Peter Saint-Andre. Extensible messaging and presence protocol (xmpp): Core. Technical report, IETF proposed standard, RFC 3920, Oct 2004.

[7] Dan Brickley and Libby Miller. FOAF Vocabulary Specification. Namespace document, FOAF Project, September 2004.

[8] Richard Newman, Danny Ayers, and Seth Russell. Tag ontology. Technical report, December 2005.

[9] Eric Prud'hommeaux and Andy Seaborne. SPARQL query language for RDF. Technical report, World Wide Web Consortium, 2005.

[10] Robert H. Warren, Dana Wilkinson, and Mike Warnecke. *Empirical Analysis of a Dynamic Social Network Built from PGP Keyrings*, volume 4503/2007 of *Lecture Notes in Computer Science*, pages 158–171. Springer Berlin / Heidelberg, April 2008.